Wisconsin Standards for Computer Science

Draft v1.0

2017 Jan 17

What is Computer Science Education?

Wisconsin defines Computer Science ("CS") as an academic discipline that encompasses the study of computers and algorithmic processes, including their principles, their hardware and software designs, their applications, networks, and their impact on society. The standards outlined in this document provide an important foundation to prepare students for post-secondary education and careers.

A Vision for Computer Science

The Wisconsin vision for CS standards is shaped by Wisconsin practitioners, experts, and the business community, and is informed by work at the national level and in other states. The Computer Science Teachers Association (CSTA) is a professional organization that supports and promotes the teaching of CS. The 2011 CSTA K–12 CS Standards represented the consensus view across the computing profession, educators and academia, and were categorized into five conceptual strands (Computational Thinking, Collaboration, Computing Practice & Programming, Computer & Communication Devices; and Community, Global & Ethical Impacts). CS and CS education are dynamic disciplines. The next community revision of the CSTA standards are expected in mid-2017, but our work in Wisconsin is informed by an interim draft made available during 2016, as well as a K-12 CS Framework under development with the involvement of many other states. The Interim 2016 CSTA K–12 CS Standards are categorized into the five concepts of the K–12 CS Framework: Computing Systems, Networks and the Internet, Algorithms and Programming, Data and Analysis, and Impacts of Computing. There is some overlap between strands and concepts, but they are not identical. The (interim) CSTA K-12 standards suggest steps that will be needed to enable their wide implementation. It is intended to introduce the principles and methodologies of CS to all students, whether they are college bound or career bound after high school. The Wisconsin vision for K-12 CS standards and the CSTA CS Standards are intended to:

1. Introduce the fundamental concepts of CS to all students, beginning at the elementary school level;
2. Present CS at the secondary school level in a way that will be both accessible and worthy of a CS credit, or as a graduation

credit;
3. Offer additional secondary-level CS standards that will allow interested students to study facets of CS in depth and prepare them for entry into a career or college; and
4. Increase the knowledge of CS for all students, especially those from under-represented groups in this field.

Computer Science Education in Wisconsin

Computer Science drives job growth and innovation throughout the economy and society. In 2016, demand for computing jobs in Wisconsin was more than three times the average demand rate for other jobs, and this trend is projected to continue for much of the next decade. The need for CS education is at an all-time high. All students need foundational knowledge in CS. To offer formal course work and integrate CS into K-12 learning opportunities, developing CS academic standards across grades K-12 is an essential first step. Not all Wisconsin school districts offer programs in CS, but all should be offering a systemic approach that prepares students to be college and career ready.

At the elementary level, CS content and concepts should be integrated throughout the curriculum. Teachers can effectively use CS concepts in instruction and activities to develop foundational skills and also can create a connection to secondary CS options. At the middle and high school levels, all students should have access to CS, including the those who wish to pursue advanced courses.

Wisconsin's Approach to Standards for Computer Science

With the release of the Wisconsin Standards for Computer Science (CS), Wisconsin CS teachers have access to the foundational knowledge and skills needed to educate students for successful entry into hundreds of high-wage, high-demand occupations and careers. Vetted by business, industry and education professionals, these standards guide Wisconsin schools, teachers and community partners toward development and continuous improvement of world-class CS courses.

The learning priorities and performance indicators contained within each set of CS standards consists of knowledge and skills specific to each of the five strands:
- Algorithms and Programming,
- Computing Systems,
- Data and Analysis,
- Impacts of Computing, and
- Networks and the Internet.

These are, of course, critical as students develop in understanding CS as a discipline as well as how these skills intersect with other content areas. In addition, there are many knowledge areas, skills and dispositions that are common to the pursuit of careers and post-secondary education in many fields.

Numerous existing sets of standards and standards-related documents have been used in developing the Wisconsin Standards for Computer Science. These include:

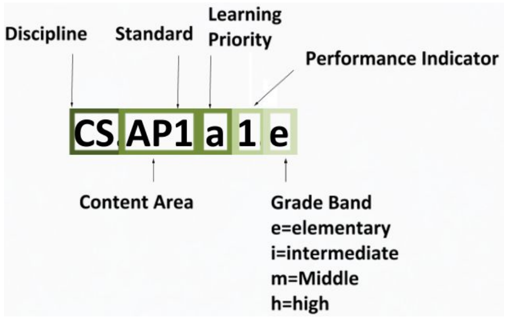
- The (Interim) CSTA K-12 Computer Science Standards, revised 2016 http://www.csteachers.org/?page=CSTA_Standards
- The K-12 CS Framework https://k12cs.org/
- Approved or draft standards from the following states:
    - Arkansas: http://www.arkansased.gov/divisions/learning-services/curriculum-and-instruction/curriculum-framework-documents/computer-science
    - Florida: http://www.cpalms.org/Public/search/Standard (under Science K-2, 3-5, 6-8, 9-12)
    - Idaho (Draft): http://www.sde.idaho.gov/topics/content-standards/index.html
    - Indiana: http://www.doe.in.gov/standards/science-computer-science
    - Massachusetts: http://www.doe.mass.edu/stem/standards.html
    - New Jersey: http://www.state.nj.us/education/cccs/2014/tech/82.pdf
    - South Carolina (draft): http://ed.sc.gov/instruction/standards-learning/computer-science/
    - Texas: http://ritter.tea.state.tx.us/rules/tac/chapter126/index.html
    - Washington: http://www.k12.wa.us/ComputerScience/LearningStandards.aspx

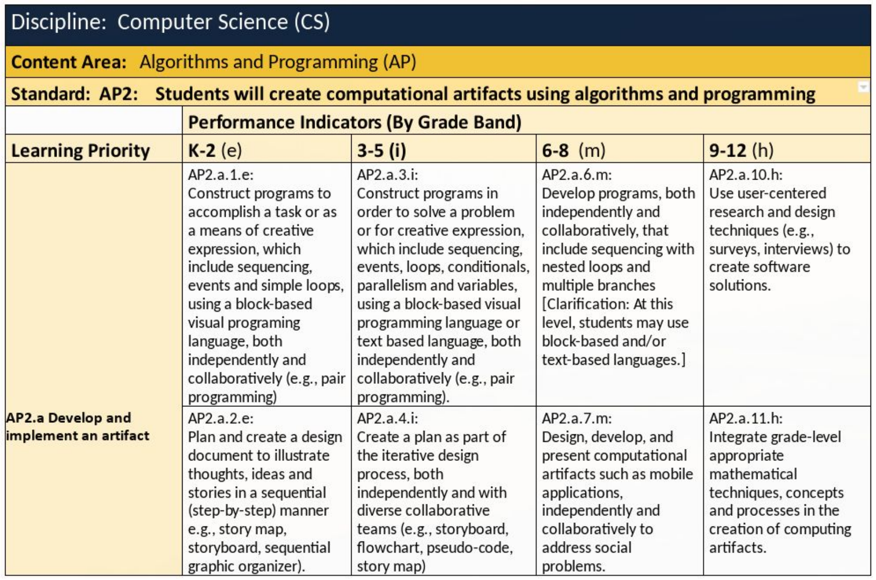
As with all the standards, the Wisconsin Standards for Computer Science may be taught and integrated through a variety of classes and experiences. Each district, school and program area should determine the means by which students meet these standards. Through the collaboration of multiple stakeholders, these foundational standards will set the stage for high-quality, successful, contemporary CS courses and programs throughout Wisconsin's PK-12 systems.

Standard Structure

The Wisconsin Standards for Computer Science follow a similar structure to all Wisconsin State Standards.

Standard Coding





Discipline: Computer Science (CS)

Content Area: Algorithms and Programming (AP)

Standard: AP2: Students will create computational artifacts using algorithms and programming

	Performance Indicators (By Grade Band)			
Learning Priority	**K-2** (e)	**3-5 (i)**	**6-8** (m)	**9-12** (h)
AP2.a Develop and implement an artifact	AP2.a.1.e: Construct programs to accomplish a task or as a means of creative expression, which include sequencing, events and simple loops, using a block-based visual programing language, both independently and collaboratively (e.g., pair programming)	AP2.a.3.i: Construct programs in order to solve a problem or for creative expression, which include sequencing, events, loops, conditionals, parallelism and variables, using a block-based visual programming language or text based language, both independently and collaboratively (e.g., pair programming).	AP2.a.6.m: Develop programs, both independently and collaboratively, that include sequencing with nested loops and multiple branches [Clarification: At this level, students may use block-based and/or text-based languages.]	AP2.a.10.h: Use user-centered research and design techniques (e.g., surveys, interviews) to create software solutions.
	AP2.a.2.e: Plan and create a design document to illustrate thoughts, ideas and stories in a sequential (step-by-step) manner e.g., story map, storyboard, sequential graphic organizer).	AP2.a.4.i: Create a plan as part of the iterative design process, both independently and with diverse collaborative teams (e.g., storyboard, flowchart, pseudo-code, story map)	AP2.a.7.m: Design, develop, and present computational artifacts such as mobile applications, independently and collaboratively to address social problems.	AP2.a.11.h: Integrate grade-level appropriate mathematical techniques, concepts and processes in the creation of computing artifacts.

Standard Formatting

- **Standard**: Broad statement that tells what students are expected to know or be able to do.
- **Learning Priority**: Breaks down the broad statement into manageable learning pieces
- **Performance Indicator by grade band**: Measurable degree to which a standard has been developed or met.

Grade Bands

Grade bands of K-2, 3-5, 6-8 and 9-12 align to typical elementary, middle and high school levels.

- Grade band K-2 and 3-5 performance indicators represent knowledge and skills that should be integrated throughout the elementary curriculum.
- Computer Science education should be part of the core curriculum for all middle school students. Awareness, exploration and building foundational skills should occur in middle school.
- Computer Science education at the high school level continues to develop student foundational understanding of CS in the world through in-depth CS learning, including awareness and exploration activities. Performance indicators marked with a "+" for grades 9-12 represent advanced CS learning expectations for students with aspirations toward careers and post-secondary studies in computing disciplines.

Content Area: Algorithms and Programming (AP)

Discipline: Computer Science (CS)				
Content Area: Algorithms and Programming (AP)				
Standard: AP1: Students will recognize and define computational problems using algorithms and programming				
	Performance Indicators (By Grade Band)			
Learning Priority	**K-2**	**3-5**	**6-8**	**9-12**
AP1.a: Develop algorithms	AP1.a.1.e: Construct and execute algorithms (sets of step-by-step instructions) that include sequencing, and simple loops to accomplish a task, both independently and collaboratively, with or without a computing device.	AP1.a.4.i: Construct and execute algorithms (sets of step-by-step instructions) which includes sequencing, loops, and conditionals to accomplish a task, both independently and collaboratively, with or without a computing device.		AP1.8.h: Design an algorithm using sequence, selection and iteration.
	AP1.a.2.e: Decompose (break down) a larger computational problem into smaller sub-problems independently or with teacher	AP1.a.5.i: Decompose (break down) a larger computational problem into smaller sub-problems independently or in a collaborative group.	AP1.a.6.m: Decompose a computational problem into parts and create solutions for one or more parts.	AP1.a.9.h: Explain and demonstrate how modeling and simulation can be used to explore natural phenomena (e.g., flocking behaviors, queueing, life cycles)

	guidance. (For example, to draw a snowman, we can draw several different, simpler shapes.)			
	AP1.a.3.e: Categorize a group of items based on the attributes or actions of each item, with or without a computing device.		AP1.a.7.m: Identify how sub-problems could be recombined to create something new (for example, break down the individual parts that would be needed to program a certain type of game and then show how the parts could be reused in other types of games.)	AP1.a.10.h: (+) Provide examples of computationally solvable problems and difficult-to-solve problems.
				AP1.a.11.h: (+) Decompose a large-scale computational problem by identifying generalizable patterns and applying them in a solution.
				AP1.a.12.h: (+) Illustrate the flow of execution of a recursive algorithm.

				AP1.a.13.h: (+) Describe how parallel processing can be used to solve large computational problems (SETI at Home, FoldIt).
				AP1.a.14.h: (+) Develop and use a series of test cases to verify that a program performs according to its design specifications.
				AP1.a.15.h: (+) Explain the value of heuristic algorithms (discovery methods) to approximate solutions for difficult to solve computational problems.
Standard: AP2: Students will create computational artifacts using algorithms and programming				
AP2.a Develop and implement an artifact	AP2.a.1.e: Construct programs to accomplish a task or as a means of creative expression, which include sequencing, events and simple loops, using a block-based visual programing	AP2.a.3.i: Construct programs in order to solve a problem or for creative expression, which include sequencing, events, loops, conditionals, parallelism and variables, using a	AP2.a.6.m: Develop programs, both independently and collaboratively, that include sequencing with nested loops and multiple branches [Clarification: At this level, students may use	AP2.a.10.h: Use user-centered research and design techniques (e.g., surveys, interviews) to create software solutions.

	language, both independently and collaboratively (e.g., pair programming).	block-based visual programming language or text based language, both independently and collaboratively (e.g., pair programming).	block-based and/or text-based languages.]	
	AP2.a.2.e: Plan and create a design document to illustrate thoughts, ideas and stories in a sequential (step-by-step) manner e.g., story map, storyboard, sequential graphic organizer).	AP2.a.4.i: Create a plan as part of the iterative design process, both independently and with diverse collaborative teams (e.g., storyboard, flowchart, pseudo-code, story map).	AP2.a.7.m: Design, develop, and present computational artifacts such as mobile applications, independently and collaboratively to address social problems.	AP2.a.11.h: Integrate grade-level appropriate mathematical techniques, concepts and processes in the creation of computing artifacts.
		AP2.a.5.i: Use mathematical operations to change a value stored in a variable.	AP2.a.8.m: Use an iterative design process (e.g., define the problem, generate ideas, build, test, and improve solutions) to solve computational problems, both independently and collaboratively.	AP2.a.12.h: Design, develop, and implement a computing artifact that responds to an event (e.g., robot that responds to a sensor, mobile app that responds to a text message, sprite that responds to a broadcast).

			AP2.a.9.m: Create variables that represent different types of data and manipulate their values.	AP2.a.13.h: (+) Decompose a computational problem by creating new data types, functions or classes.
				AP2.a.14.h: (+) Develop programs for multiple computing platforms (e.g., computer desktop, web, mobile).
				AP2.a.15.h: (+) Implement an AI algorithm to play a game against a human opponent or solve a problem.
				AP2.a.16.h: (+) Demonstrate code reuse by creating programming solutions using libraries and APIs. (e.g., graphics libraries, maps API).
Standard: AP3: Students will be able to communicate about computing ideas				
AP3.a Recognize and cite sources	AP3.a.1.e: Give credit to the source when using code, music, or pictures (for	AP3.a.2.i: Use proper citations and document when ideas are borrowed and changed for their	AP3.a.3.m: Provide proper attribution when code is borrowed or built upon.	AP3.a.4.h: Compare and contrast various software licensing schemes (e.g.,

	example) that were created by others.	own use (e.g.,, using pictures created by others, using music created by others, remixing programming projects).		open source, freeware, commercial).
AP3.b Communicate about technical and social issues	AP3.b.1.e: Follow simple instructions to complete a task. For example, a simple visual tutorial.	AP3.b.2.i: Remember basic concepts/facts regarding security issues with general computer use.	AP3.b.6.m: Understand security issues with general computer use	AP3.b.10.h: (+) Explain security issues that might lead to compromised computer programs (e.g., circular references, ambiguous program calls, lack of error checking and field size checking).
		AP3.b.3.i: Understand that technology has impacted society in both beneficial and harmful ways.	AP3.b.7.m: Discuss how technology has impacted society - both the beneficial and harmful effects.	AP3.b.11.h: Evaluate and analyze how technology has impacted our society and discuss the benefits and harmful impacts of a variety of innovations in technology.
		AP3.b.4.i: Compare different problem solving techniques	AP3.b.8.m: Compare different algorithms that may be used to solve the same problem in terms of their speed, clarity and size (e.g., different algorithms solve the	AP3.b.12.h: (+) Compare a variety of programming languages and identify features that make them useful for solving different types of problems and developing different

			same problem, but one might be faster than the other). [Clarification: Students are not expected to quantify these differences.]	kinds of systems (e.g., declarative, logic, parallel, functional, compiled, interpreted, real-time).
		AP3.b.5.i: Modify a set of instructions (for example, in dance, cooking, etc) and discuss how many paths can lead to the same result.	AP3.b.9.m: Modify existing code to change its functionality, and discuss the variety of ways in which to do this.	AP3.b.13.h: (+) Modify an existing program to add additional functionality and discuss intended and unintended implications (e.g., breaking other functionality).
AP3.c Document code			AP3.c.1.m: Interpret the flow of execution of algorithms and predict their outcomes. [Clarification: Algorithms can be expressed using natural language, flow and control diagrams, comments within code, and pseudocode).	AP3.c.3.h: (+) Describe how artificial intelligence drives many software and physical systems (e.g., autonomous robots, computer vision, pattern recognition, text analysis).
			AP3.c.2.m: Use documentation regarding code to modify programs	AP3.c.4.h: Write appropriate documentation for programs

				AP3.c.5.h: Use application programming interface (APIs) documentation resources.
				AP3.c.6.h: Use online resources to answer technical questions

Standard: AP4: Students will develop and use abstractions

AP4.a Create and use abstractions (representations) to solve complex computational problems	AP4.a.1.e: Use numbers or other symbols to represent data (e.g., thumbs up/down for yes/no, color by number, arrows for direction, encoding/decoding a word using numbers of pictographs).	AP4.a.2.i: Use several existing functions/procedures to solve a problem (e.g., using several square, circle and triangle drawing functions to create a larger picture).	AP4.a.3.m: Define and use functions/procedures that hide the complexity of a task and can be reused to solve similar tasks. [Clarification: Students use and modify, but do not necessarily create, functions/procedures with parameters.]	AP4.a.4.h: Demonstrate the value of abstraction for managing problem complexity (e.g., using a list instead of discrete variables).
				AP4.a.5.h: Understand the notion of hierarchy and abstraction in high-level languages, translation, instruction sets, and logic circuits.

				AP4.a.6.h : Deconstruct a complex problem into simpler parts using predefined constructs (e.g., functions and parameters and/or classes).
				AP4.a.7.h: (+) Compare and contrast fundamental data structures and their uses (e.g., lists, maps, arrays, stacks, queues, trees, graphs).
				AP4.a.8.h: (+) Critically analyze and evaluate classic algorithms (e.g., sorting, searching) and use in different contexts, adapting as appropriate.
				AP4.a.9.h: (+) Discuss issues that arise when breaking large-scale problems down into parts that must be processed simultaneously on separate systems (e.g., cloud computing,

				parallelization, concurrency).
				AP4.a.10.h: (+) Define the functionality of an abstraction without implementing the abstraction.
				AP4.a.11.h: (+) Evaluate algorithms (e.g., sorting, searching) in terms of their efficiency, correctness and clarity.
				AP4.a.12.h: (+) Identify programming language features that can be used to define or specify an abstraction.
				AP4.a.13.h: (+) Identify abstractions used in a solution (program or software artifact) and reuse those abstractions to solve a different problem.
Standard: AP5: Students will be able to collaborate with diverse teams				
AP5.a Use a variety of resources to work together to solve computational problems	AP5.a.1.e: Work together with a team to create a	AP5.a.3.i: Apply collaboration strategies to support	AP5.a.5.m: Solicit and integrate peer feedback as	AP5.a.6.h:

	solution to a computational problem.	problem solving within the design cycle of a program.	appropriate to develop or refine a program.	Design and develop a software artifact working in a team.
	AP5.a.2.e: Use teachers, parents, and other resources to solve a computational problem.	AP5.a.4.i: Understand there are many resources that can be used/tapped to solve a problem.		AP5.a.7.h: Demonstrate how diverse collaborating impacts the design and development of software products (e.g., discussing real-world examples of products which have been improved through having a diverse design team or reflecting on their own team's development experience).
				AP5.a.8.h: (+) Demonstrate software life cycle processes (e.g., spiral, waterfall) by participating on software project teams (e.g., community service project with real-world clients).

				AP5.a.9.h: (+) Use version control systems, Integrated Development Environments (IDEs), and collaboration tools and practices (code documentation) in a group software project.
AP5.b Fostering an inclusive computing culture	AP5.b.1.e: Understand the value that all members bring to the table with difference of gender, race, religion,etc.		AP5.b.2.m: Analyze team members strengths and utilize them.	AP5.b.3.h: Create design teams taking into account the strengths and perspectives of potential team members.
Standard: AP6: Students will be able to test and refine computational solutions				
AP6.a Test and debug computational solutions	AP6.a.1.e: Analyze and debug (fix) an algorithm that includes sequencing and simple loops, with or without a computing device.	AP6.a.2.i: Analyze and debug (fix) an algorithm which includes sequencing, events, loops, conditionals, parallelism, and variables.	AP6.a.3.m: Use testing and debugging methods to ensure program correctness and completeness.	AP6.a.4.h: Use a systematic approach and debugging tools to independently debug a program (e.g., setting breakpoints, inspecting variables with a debugger).

AP6.b Develop and apply success criteria		AP6.b.1.i: Determine the correctness of a computational problem solution by listening to a classmate describe the solution.	AP6.b.2.m: Apply a rubric to determine if and how well a program meets objectives	AP6.b.3.h: (+) Evaluate key qualities of a program (e.g., correctness, usability, readability, efficiency, portability, scalability) through a process such as a code review.

Content Area:
Computing Systems (CS)

Discipline: Computer Science (CS)				
Content Area: Computing Systems (CS)				
Standard: CS1: Students will communicate about computing systems				
	Performance Indicators (By Grade Band)			
Learning Priority	**K-2** (e)	**3-5 (i)**	**6-8** (m)	**9-12** (h)
CS1.a: Identification of hardware and software components	CS1.a.1.e: Identify and use software that controls computational devices to accomplish a task (e.g., use an app to draw on the screen, use software to write a story or control robots).		CS1.a.4.m: Justify the suitability of hardware and software chosen to accomplish a task (e.g., comparison of the features of a tablet vs. desktop, selecting which sensors and platform to use in building a robot or developing a mobile app).	CS1.a.5.h: Develop and apply criteria (e.g., power consumption, processing speed, storage space, battery life, cost, operating system) for evaluating a computer system for a given purpose (e.g., system specification needed to run a game, web browsing, graphic design or video editing).
	CS1.a.2.e: Use appropriate terminology in naming and describing the function of common computing devices and	CS1.a.3.i: Use appropriate terminology in naming internal and external components of computing devices and		CS1.a.6.h: (+) Identify the functionality of various categories of hardware components and communication

	components (e.g., desktop computer, laptop computer, tablet device, monitor, keyboard, mouse, printer).	describing their relationships, capabilities, and limitations.		between them (e.g., physical layers, logic gates, chips, input and output devices).
CS1.b: Understand how the components of a computer system work together	CS1.b.1.e: Identify the components of a computer system and what the basic functions are (hard drive, memory, etc.) as well as external features and their uses (printers, scanners, external hard drives, etc.) For lower elementary students, common hardware like iPads, Kindles, or Chromebooks can be used.	CS1.b.2.i: Model how a computer system works. [Clarification: Only includes basic elements of a computer system, such as input, output, processor, sensors, and storage.]		CS1.b.3.h: (+) Explain the role of operating systems (e.g., how programs are stored in memory, how data is organized/retrieved, how processes are managed and multi-tasked).
Standard: CS2: Students will test and refine computing systems				
CS2.a: Problem solving, trouble-shooting, debugging	CS2.a.1.e: Identify, using accurate terminology, simple hardware and software problems that may occur during use (e.g., app or program not working as expected, no	CS2.a.2.i: Identify, using accurate terminology, simple hardware and software problems that may occur during use, and apply strategies for	CS2.a.3.m: Use a systematic process to identify the source of a problem within individual and connected devices (e.g., follow a troubleshooting flow diagram, make changes to	

	sound, device won't turn on).	solving problems (e.g., reboot device, check for power, check network availability, close and reopen app).	software to see if hardware will work, restart device, check connections, swap in working components).	
Standard: CS3: Students will develop and use abstractions in computing systems				
CS3.a: Generalization in computer systems			CS3.a.1.m: Analyze the relationship between a device's computational components and its capabilities. [Clarification: Computing Systems include not only computers, but also cars, microwaves, smartphones, traffic lights, and flash drives.]	CS3.a.2.h: Demonstrate the role and interaction of a computer embedded within a physical system, such as a consumer electronic, biological system, or vehicle, by creating a diagram, model, simulation, or prototype.
				CS3.a.3.h: Describe the steps necessary for a computer to execute high-level source code (e.g., compilation to machine language, interpretation, fetch-decode-execute cycle).

Standard: CS4: Students will create and modify computing systems				
CS4.a: Modifying and creating artifacts			CS4.a.1.m: Extend or modify existing programs to add simple features and behaviors using different forms of inputs and outputs (e.g., inputs such as sensors, mouse clicks, data sets; outputs such as text, graphics, sounds).	CS4.a.2.h: Create, extend, or modify existing programs to add new features and behaviors using different forms of inputs and outputs (e.g., inputs such as sensors, mouse clicks, data sets; outputs such as text, graphics, sounds).
				CS4.a.3.h: (+) Create a new artifact that uses a variety of forms of inputs and outputs (e.g., inputs such as sensors, mouse clicks, data sets; outputs such as text, graphics, sounds).

Content Area:
Data and Analysis (DA)

Discipline: Computer Science (CS)				
Content Area: Data and Analysis (DA)				
Standard: DA1: Students will create computational artifacts using data and analysis				
	Performance Indicators (By Grade Band)			
Learning Priority	**K-2**	**3-5**	**6-8**	**9-12**
DA1.a: Techniques for representing, and manipulating data.		DA1.a.1.i: Use numeric values to represent non-numeric ideas in the computer (binary, ASCII, pixel attributes such as RGB).	DA1.a.3.m: Represent data using different encoding schemes (e.g., binary, Unicode, Morse code, shorthand, student-created codes).	DA1.a.5.h: Convert between binary, decimal, and hexadecimal representations of data (e.g., convert hexadecimal color codes to decimal percentages, ASCII/Unicode representation).
		DA1.a.2.i: Answer a question by using a computer to manipulate (e.g., sort, total and/or average, chart, graph) and analyze data that has been collected by the class or student.	DA1.a.4.m: Revise computational models to more accurately reflect real-world systems (e.g., ecosystems, epidemics, spread of ideas).	DA1.a.6.h: Create computational models that simulate real-world systems (e.g., ecosystems, epidemics, spread of ideas).

				DA1.a.7.h: (+) Discuss how data sequences (e.g., binary, hexadecimal, octal) can be interpreted in a variety of forms (e.g., instructions, numbers, text, sound, image).
Standard: DA2: Students will recognize and define data in computational problems				
DA2.a: Categorizing and analyzing data	DA2.a.1.e: Sort objects into buckets, recognizing relevant and/or irrelevant data (i.e. one of these things is not like the other)	DA2.a.2.i: Choose appropriate classifications or grouping for data by shape, color, size, or other attributes.	DA2.a.3.m: Develop a strategy to answer a question by using a computer to manipulate (e.g., sort, total and/or average, chart, graph) and analyze data that has been collected by the class or student.	DA2.a.4.h: Apply basic techniques for locating and collecting small- and large-scale data sets (e.g., creating and distributing user surveys, accessing real-world data sets).
DA2.b: Gathering data				DA2.b.1.h: Discuss techniques used to store, process, and retrieve different amounts of information (e.g., files, databases, data warehouses).
				DA2.b.2.h: (+) Use various data collection techniques

				for different types of computational problems (e.g., mobile device GPS, user surveys, embedded system sensors, open data sets, social media data sets).
Standard: DA3: Students will communicate about data in computing				
DA3.a: Communicating	DA3.a.1.e: Collect data over time and organize it in a chart or graph in order to make a prediction.		DA3.a.2.m: Describe how different formats of stored data represent tradeoffs between quality and size. [Clarification: compare examples of music, text and/or image formats.]	DA3.a.4.h: Use computational tools to collect, transform, and organize data about a problem to explain to others.
			DA3.a.3.m: Explain the processes used to collect, transform, and analyze data to solve a problem using computational tools (e.g., use an app or spreadsheet form to collect data, decide	

			which data to use or ignore, and choose a visualization method).	
Standard: DA4: Students will develop and use data abstractions				
DA4.a: Modelling	DA4.a.1.e: Use a computing device to store, search, retrieve, modify, and delete information and define the information stored as data.	DA4.a.3.i: Create a computational artifact to model the attributes and behaviors associated with a concept (e.g., solar system, life cycle of a plant).		DA4.a.4.h: Analyze the representation tradeoffs among various forms of digital information (e.g., lossy vs. lossless compression, encrypted vs. unencrypted, various image representations).
	DA4.a.2.e: Create a model of an object or process in order to identify patterns and essential elements (e.g., water cycle, butterfly life cycle, seasonal weather patterns).			DA4.a.5.h: (+) Evaluate the ability of models and simulations to formulate, refine, and test hypotheses.
DA4.b: Patterns				DA4.b.1.h: (+)Use data analysis to identify significant patterns in complex systems (e.g., take

				existing data sets and make sense of them).
				DA4.b.2.h: (+) Identify mathematical and computational patterns through modeling and simulation (e.g., regression, Runge-Kutta, queueing theory, discrete event simulation).

Content Area:
Impacts of Computing (IC)

Discipline: Computer Science (CS)				
Content Area: Impacts of Computing (IC)				
Standard: IC1: Students will understand the impact technology has on our everyday lives by describing the beneficial and harmful effects computing innovations have had and will have on our economy, our culture, and our relationships.				
	Performance Indicators (By Grade Band)			
Learning Priority	**K-2**	**3-5**	**6-8**	**9-12**
IC1.a: Understand the effects of computing on the economy and culture.	IC1.a.1.e: Compare and contrast examples of how computing technology has changed and improved the way people live, work, and interact.	IC1.a.2.i: Evaluate and describe the positive and negative impacts of the pervasiveness of computers and computing in daily life (e.g., downloading videos and audio files, electronic appliances, wireless Internet, mobile computing devices, GPS systems, wearable computing).	IC1.a.4.m: Provide examples of how computational artifacts and devices impact health and wellbeing, both positively and negatively.	IC1.a.6.h: Debate the social and economic implications associated with ethical and unethical computing practices (e.g., intellectual property rights, hacktivism, software piracy, new computers shipped with malware).

		IC1.a.3.i: Generate examples of how computing can affect society, and also how societal values can shape computing choices.	IC1.a.5.m: Explain how computer science fosters innovation and enhances nearly all careers and disciplines.	IC1.a.7.h: Discuss implications of the collection and large-scale analysis of information about individuals (e.g., how businesses, social media, and government collect and use personal data).
				IC1.a.8.h: Compare and debate the positive and negative impacts of computing on behavior and culture (e.g., evolution from hitchhiking to ride-sharing apps, online accommodation rental services).
				IC1.a.9.h: Describe how computation shares features with art and music by translating human intention into an artifact.
				IC1.a.10.h: (+) Develop criteria to evaluate the beneficial and harmful effects of computing innovations on people and society.

IC1.b: Understand the effects of computing on communication and relationships	IC1.b.1.e: Explain the differences between communicating electronically and communicating in person.	IC1.b.2.i: Compare and contrast the effects of communicating electronically to communicating in person.	IC1.b.3.m: List beneficial and harmful effects of personal electronic communication and social electronic communication.	IC1.b.5.h: Evaluate the negative impacts of electronic communication on personal relationships and evaluate differences between face-to-face and electronic communication.
				IC1.b.6.h: (+) Create a list of practices that individuals and organizations can use to encourage proper use of both electronic and face-to-face communication.
			IC1.b.4.m: Describe ways in which the Internet impacts global communication and collaborating.	IC1.b.7.h: (+) Evaluate the negative impacts on societal discourse caused by social media and electronic communities.
Standard: IC2: Students will experience learning within a collaborative, inclusive computing culture and explain the steps needed to ensure that all people have access to computing.				
IC2.a Understand the effects of the Digital Divide		IC2.a.1.i: Brainstorm ways in which computing devices and the Internet could be made more	IC2.a.2.m: Explain the impact of the digital divide (i.e., uneven access to computing, computing education,	IC2.a.3.h: (+) Evaluate the impact of equity, access, and influence on the distribution of computing

		available to all people.	and interfaces) on access to critical information.	resources in a global society.
IC2.b: Test and refine digital artifacts for accessibility		IC2.b.1.i: Brainstorm ways in which computing devices could be made more accessible to all users.	IC2.b.2.m: Critically evaluate and redesign a computational artifact to remove barriers to universal access (e.g., using captions on images, high contrast colors, and/or larger font sizes).	IC2.b.3.h: Design a user interface (e.g., webpages, mobile applications, animations) to be more inclusive, accessible, and minimizing the impact of the designer's inherent bias.
IC2.c: Collaborate in the creation of digital artifacts		IC2.c.1.i: Use the Internet to work with another student to solve a problem or reach a goal.	IC2.c.3.m: Use the Internet to work with a group of people to solve a problem or reach a goal who are not physically near.	IC2.c.4.h: Select, observe, and contribute to global collaboration in the development of a computational artifact (e.g., contribute the resolution of a bug in an open-source project hosted on GitHub, or contribute to a Wikipedia article).
		IC2.c.2.i: Seek out and compare diverse perspectives, synchronously or asynchronously, to improve a project.		IC2.c.5.h: Demonstrate how computing enables new forms of experience, expression, communication, and collaboration.

Standard: IC3: Students will understand the importance of proper use of data and information in a computing society.				
IC3.a Understand intellectual property and fair use		IC3.a.1.i: Use resources from the World Wide Web in making artifacts and recognize that the work came from others.	IC3.a.2.m: Understand laws associated with digital information such as intellectual property, fair use, and Creative Commons.	IC3.a.4.h: Compare and contrast information access and distribution rights.
			IC3.a.3.m: Describe ethical issues that relate to computing devices and networks (e.g., equity of access, security, hacking, intellectual property, copyright, Creative Commons licensing, and plagiarism).	
IC3.b Assess the practice of digital privacy	IC3.b.1.e: Respect other students' information and refrain from accessing others' devices or accounts without permission.	IC3.b.2.i: Understand what kinds of digital information is considered private, take steps to keep their information private, and respect the privacy of other	IC3.b.4.m: Summarize negative and positive impacts of using data and information to categorize people, predict behavior, and make recommendations based on those predictions (e.g., customizing search results or targeted	IC3.b.5.h: Investigate misuses of private digital information in our society.

		students' information.	advertising, based on previous browsing history, can save search time and limit options at the same time).	
		IC3.b.3.i: Explain problems that relate to using computing devices and networks (e.g., logging out to deter others from using your account, cyberbullying, privacy of personal information, and ownership).		IC3.b.6.h: Debate laws regarding an individual's digital privacy and be able to explain the main arguments from each side.
IC3.c: Assess interrelationship between computing and society				IC3.c.1.h: (+) Design and implement a study that evaluates how computation has revolutionized an aspect of our culture, or predicts how an aspect might evolve (e.g., education, healthcare, art/entertainment, energy).
				IC3.c.2.h: (+) Debate laws and regulations that impact

				the development and use of software.

Content Area: Networking and the Internet (NI)

Discipline: Computer Science (CS)				
Content Area: Networking and the Internet (NI)				
Standard: NI1: Students will understand the importance of security when using technology				
	Performance Indicators (By Grade Band)			
Learning Priority	**K-2**	**3-5**	**6-8**	**9-12**
NI1.a: Use secure practices for personal computing	NI1.a.1.e: Use secure practices (such as passwords) to protect private information and discuss the effects of misuse.	NI1.a.2.i: Create examples of strong passwords, explain why strong passwords should be used, and demonstrate proper use and protection of personal passwords.	NI1.a.3.m: Summarize security risks associated with weak passwords, lack of encryption, insecure transactions, and persistence of data.	NI1.a.4.h: Provide examples of personal data that should be kept secure and the methods by which individuals keep their private data secure.
NI1.b: Understand the importance of institutional security		NI1.b.1.i: Give examples of information that organizations keep private as opposed to information that they make public.	NI1.b.2.m: Explain the principles of information security (confidentiality, integrity, availability) and authentication techniques.	NI1.b.3.h: Compare and contrast multiple viewpoints on cybersecurity (e.g., from the perspective of security experts, privacy advocates, the government).

				NI1.b.4.h: Identify digital and physical strategies to secure networks and discuss the tradeoffs between ease of access and need for security.
Standard: NI2: Students will understand how information is sent by the internet				
NI2.a: Demonstrate how the internet works at the physical layer	NI2.a.1.e: Use a physical tool (e.g. flashlight, string) to communicate with another student.	NI2.a.3.i: Model how a device on a network sends a message from one device (sender) to another (receiver) while following specific rules.	NI2.a.6.m: Simulate how information is transmitted as packets through multiple devices over the internet and networks.	NI2.a.8.h: Illustrate the basic components of computer networks (e.g., draw logical and topological diagrams of networks including routers, switches, servers, and end user devices; create model with string and paper).
	NI2.a.2.e: Differentiate between using the internet and not using the internet (i.e. identify difference between local and remote computation)	NI2.a.4.i: Provide examples of computer use that involves the Internet.	NI2.a.7.m: Explain using basic terms how a wi-fi or cellular network allows Internet information to be transmitted from a server to their device.	NI2.a.9.h: (+) Explain ways in which the Internet is decentralized and fault-tolerant.

		NI2.a.5.i: Explain in a basic way how Internet information arrives at a computer.		NI2.a.10.h: (+) Simulate and discuss the issues (e.g., bandwidth, load, delay, topology) that impact network functionality (e.g., use free network simulators).
NI2.b: Demonstrate how the internet works at the protocol layer		NI2.b.1.i: Act out a protocol that people use in common everyday communications (such as checking out a book from the library, meeting a new person, making an appointment, playing a class game, or calling a friend on the phone to invite them over.	NI2.b.2.m: Define the term protocol. provide an example of protocols in daily life, and explain their use on the Internet.	NI2.b.3.h: Describe key protocols and underlying processes of Internet-based services (e.g., http/https and SMTP/IMAP, routing protocols).
NI2.c: Demonstrate how the internet works at the addressing layer	NI2.c.1.e: Devise a system for sending a physical message to anyone in their school by using addressing techniques (e.g., address valentine envelopes by	NI2.c.2.i: Devise a system for sending a physical message to anyone in their school by using addressing techniques, and then draw a tree or visual	NI2.c.3.m: Explain the hierarchical structure of the Internet Domain System.	NI2.c.4.h: (+) Evaluate how the hierarchical nature of the Domain Name System helps the Internet work efficiently.

	student first name, and teacher, grade, or room).	representation of their addressing system, and finally act out their addressing system by sending messages.		
NI2.d: Demonstrate and explain encryption methods		N12.d.1.i: Secretly communicate across a classroom using a method of their own design (e.g., pictures, physical movement, text).	NI2.d.2.m: Encode and decode text-based messages using basic algorithms (e.g., shift cipher, substitution cipher).	NI2.d.3.h: Write a program that performs basic encryption (e.g., shift cipher, substitution cipher).
				NI2.d.4.h: (+) Explain the features of public key cryptography.
				NI2.d.5.h: (+) Explore security policies by implementing and comparing encryption and authentication strategies (e.g., secure coding, safeguarding keys).